

DATA MODEL FOR SUPPLY CHAIN PLANNING

Background of the invention

Supply chain planning, which comprises the logistical plan of an in-house supply chain, is essential to the success of many of today's manufacturing firms. Most manufacturing firms rely on supply chain planning in some form to ensure the timely delivery of products in response to customer demands. Typically, supply chain planning is hierarchical in nature, extending from distribution and production planning driven by customer orders, to materials and capacity requirements planning, to shop floor scheduling, manufacturing execution, and deployment of products. Supply chain planning ensures the smooth functioning of different aspects of production, from the ready supply of components to meet production demands to the timely transportation of finished goods from the factory to the customer.

The modern supply chain often encompasses a vast array of data. The planning applications that create and dynamically revise plans in the supply chain in response to changing demands and capacity require rapid access to data concerning the flow of materials through the supply chain. The efficient operation of the supply chain depends upon the ability of the various plans to adjust to changes, and the way in which the required data is stored determines the ease with which it can be accessed.

In the conventional relational model, supply chain data is stored in multiple relational database tables. If a parameter of a manufacturing order is changed, all of the

aspects of the supply chain affected by such change must be re-calculated using the relational tables. Before a planning algorithm can change the date and/or quantity of a manufacturing order in response to changing capacities, for example, it must take into account the effect that the date and/or quantity change will have on other production and sales orders. Such a calculation is very complex, and requires that the algorithm have access to data concerning all the other orders, materials and resources that would be affected by the change. That information is not readily accessible in the conventional model, and instead must be calculated by tracing through relational database tables. Such calculations are cumbersome and delay planning functions unnecessarily.

15 There is therefore a need for all data relevant to supply chain planning to be stored in an efficient manner which reflects the progress of materials and orders along the supply chain. There is also a need for such data to be made available to planning algorithms in the most efficient and

20 usable manner possible so as to reduce drastically the runtime of the planning functions.

Summary of the invention

It is a further object of this invention to allow a planning algorithm efficient access to any organized group of working steps in the production process by providing a database table whereby each of the groups of working steps is
5 referenced to its location in the data structure.

It is a further object of this invention to allow a planning algorithm efficient access to the working step performed by a given resource at a specific time, by providing a database table whereby the dates and times of all working steps
10 performed by each resource are referenced to that resource.

It is a further object of the invention to allow a planning algorithm to have efficient access to organized groups of working steps involved in creating or consuming a specific material, by providing a database table whereby information
15 identifying the material is referenced to the input or output of each such organized group of working steps.

In accordance with these and other objects, a data structure is defined whereby individual working steps in the production process are defined as activities, and organized
20 groups of such activities are defined as orders. Activities are allocated to no more than one resource, if any, and contain information concerning the start and finish time for the activity, any resource on which the activity is currently scheduled, and a list of alternative resources, if
25 any. Activities representing a time calculation only are

not required to correspond to a resource. Activities are linked to each other via auxiliary objects, which contain information concerning the minimum and maximum time between activities. Orders may contain input and/or output

5 interface nodes, representing the materials consumed and produced by activities within the order. An output interface node representing a quantity of material created from one order is linked via an auxiliary object to

10 respective input interface node or nodes from other orders that require that material. Order anchors are defined whereby a planning algorithm can easily reference an order in the data structure by its order number in a database table. Planning object anchors allow the planning algorithm to access all the orders for a given material, and resource

15 anchors permit access to all activities scheduled for that resource.

Brief Description of the Drawings

Figure 1 is a representation of the relationship

20 between an order and its activities.

Figure 2 is a representation of the fusing of activities to make operations.

Figure 3 is a representation of the input and output interface nodes of an order.

SECRET

Figure 6 is a representation of how temporal constraints between orders are modeled.

Figure 8 is a representation of how pegging
10 between orders is modeled.

Figure 10 is a representation of a planning object anchor.

DETAILED DESCRIPTION OF THE DRAWINGS

6

points to activity 12 and activity 13. Each activity 11 contains a reference 14 to its corresponding orders. As depicted in Figure 2, related activities such as a chain of activities 11a, 11b, and 11c that must be executed in order
5 may be grouped together into an "operation" 20 to avoid having to map each activity individually on a planning table.

An order, such as order 10, may have one or more input interface nodes 30 and/or one or more output interface
10 nodes 31, as shown in Figure 3. Each input interface node 30 represents one material. An input interface node also has attributes containing information as to the quantity of the material required, the time requirement of the material, and the shortage of that material, which is derived from the
15 difference between the quantity of material required and the quantity that it delivered by other orders or stock. Each output interface node 31 has similar attributes, such as type of material created, the quantity of the material, the time availability of the material, and the surplus of that
20 material, which is derived from the quantity of material produced that is not yet delivered to other orders. Each input interface node 30 may refer to the activity 12, if any, in which the material that it represents is consumed, and each output interface node 31 points to the activity 13,
25 if any, in which the material that it represents is created. If an activity 12 consumes a material, all input

materials of this activity can be traced via arrow marked
with dashes and dots 32a that points from activity 12 to
input interface node 30. If input activity 12 consumes more
than one material, arrow 32b joins input interface node 30a
5 to the next input interface node 30b, which links on the
same input activity 12. The chain of input interface nodes
30, which can have an infinite length, enables the user to
ascertain which materials are required for the order in
question, which means that he can determine the bill of
10 materials for the output materials. Similarly, the output
materials of an activity can be traced via arrow marked with
dashes and dots 34 joining the activity with the first
output interface node 31 of the activity 13. If there are
several output materials, then an arrow 33 joins output
15 interface node 31a to the next output interface node 31b,
which represents the second material created. Like the
input interface nodes 30, this chain can have an infinite
length.

Figure 4 illustrates order 10 having activities 41
20 with no input or output materials, and also activities such
as assembly process 42 that both consume and create
materials. Stock or a purchase order of a material is
modeled by an order with one output interface node 31 and
with no input interface nodes or activities. If the user
25 wants to take the capacity of the vendor into account in
modeling a purchase order, however, the purchase order must

contain at least one activity representing the available capacity of the vendor (as accurately as possible). A plurality of purchase orders may be grouped into a "schedule line", which is modeled as a purchase order with several
5 output interface nodes, each with a different delivery time. Customer requirements mirror purchase orders: each order has one input interface node 30, but no output interface nodes or activities. Orders that consume materials such as customer orders are modeled as "issuing elements", while
10 orders that create materials are modeled as "receiving elements", allowing both types of orders to be modeled as objects similar to other components in the SCP model, thereby saving the need to create additional algorithms to operate on the model. An order without input or output
15 interface nodes may exist, for example, as an order representing a test or maintenance on a resource.

Links may also be created between successive activities, said links containing references not only to successor activities, but also to the minimum and maximum
20 time between activities. These temporal constraints can exist both between activities in the same order (inter-order constraints), and between activities belonging to different orders (cross-order constraints).

Figure 5 illustrates inter-order temporal
25 constraints between activities. As shown in Figure 5a, activity 11d has three successor activities 11e, 11f, and

11g. Edges 50, 51 and 52 representing the temporal constraints have attributes, which are the minimum and maximum time interval between activities, and the type of temporal constraint, such as start-start, start-finish, finish-finish, or finish-start. References to successive activities are modeled by following the full and dotted straight arrows. In Figure 5b, starting from activity 11d, first follow the arrow "succ_edge" 51 to reach the first successor activity from the small square 53 along the edge "succ_act" 52. This process is repeated from the first small square in order to reach all other successor activities successively. These small squares 53, also known as auxiliary objects, store the references to the successor activities, the next auxiliary object, and all the attributes of the temporal constraint between activities. The same technique can be used to model predecessor constraints, for example by following the curved arrows "pred_edge" 54 and "pred_act" 55 to find the predecessors of activity 11h.

Cross-order temporal constraints are illustrated in Figure 6. A first order 61 and a second order 62 each contain three activities 11. There is a cross-order temporal constraint between activity 11j of first order 61 and activity 11n of second order 62. This constraint is mapped in the same way as described above for an inter-order temporal constraint, showing that activity 11j has successor

activities 11k and 11o, and that activity 11o has predecessor activities (11j and 11n).

Similarly, "pegging" links two orders wherein one of the orders supplies a material consumed by the other order. Pegging tracks the type and quantity of material supplied by one order (the "subordinate order") to another order (the "superior order"). Pegging allows the planner to ascertain the superior and subordinate orders for any given order at any given time. If the planner reschedules the dates of an order, pegging allows all other orders influenced by this change to be updated.

Figure 7 illustrates an example of pegging between orders, consisting of five orders 71, 72, 73, 74 and 75 that produce or consume materials M1, M2 and/or M3. As shown, for example, one piece each of M2 and M3 is required to produce M1. Next to each input interface node 30 is the required quantity and the requirements date, and next to each output interface 31 node is the quantity created and the availability date. For example, order 71 produces 100 M2, which is sufficient to satisfy the demands of orders 72 and 73 producing M1. There are also two separate orders 71 and 75 satisfying the demand of order 73.

As shown in Figure 8, relationships between orders are mapped in the same manner as are temporal constraints between activities. The orders which M2 order 71 supplies

can be found by starting from output interface node 31 of the M2 order 71 and alternately following the full straight arrows ("succ_edge") 51 and the dotted straight arrows ("succ_act") 52. Similarly, the orders that supply M1 order 5 73 can be found by starting from the input interface nodes 30 of M1 order 73, and alternately following the full curved arrows ("pred_edge") 54 and dotted curved arrows ("pred_output") 55 to output interface node 31 of the supplying orders 71 and 75.

10 While pegging can link a large network of orders, not all orders have relationships to each other. Accordingly, the SCP network is usually a collection of disjunctive sub-networks, making it difficult to scan the whole network to locate a specific order. Accordingly, an 15 "order anchor" 90 as shown in Figure 9 can provide direct access to an order 10 or group of orders in the SCP network via an "order number" 91. This information is preferably stored in a RAM-buffered relational database table with a primary index for the order number and a secondary index for 20 the reference to the order in the network, or object identity ("OID") 92. Similarly, an "operation anchor" 93 can provide direct access to an operation 20, or fused activities, within an order 10. The key 97 of the operation anchor 93 is the OID 92 for order 10, which is referenced to 25 the predetermined operation number 94, an operation number within that order 95, and the first activity 96 of the

respective operation 20. So for an application program to access a certain operation 20 of an order 10, it must first use the order anchor 90 to determine the OID 92 for the order, then use the operation anchor 93 to find the first
5 activity 96 of the operation 20 that it is seeking.

Planning object anchors 100, illustrated in Figure 10, enable an application program to determine efficiently all the orders 10 for a given material. Each material is identified in a relational database table according to its
10 material number 101, plant 102, storage location 103, and batch 104, collectively known as a "planning object".

Planning object anchors in table 109 reference to first input interface node 105 for material M4, and first output interface node 106 for material M1. All input interface
15 nodes 30 and output interface nodes 31 of each material are kept in doubly concatenated lists which are sorted according to requirements and availability dates, as shown in Figure 10. Thus, it is possible to select all receiving and issuing elements for each material or "planning object",
20 which can be important for materials requirements planning.

Since each activity preferably corresponds to one specific resource, a resource anchor is provided to enable an application program to determine all activities for a specific resource. Figure 11 depicts a resource anchor 110
25 modeled as a relational database table 119 that references each resource number 111 to the corresponding first activity

A2 scheduled on that resource. The first activity A2 112 is then linked to the next activity A3 113 scheduled on the same resource, which is in turn linked to subsequent activities A5 115 and A8 118 in chronological order according to schedule time. This facilitates the scheduling of new activities on particular resources. For example, in order to schedule a new activity on a resource, an applications program must first check the activity immediately before and the activity immediately after the proposed time for the new activity to determine whether there is sufficient time to perform the new activity on that resource. Since resource anchor 110 stores all activities of a particular resource chronologically referenced to that resource, this information is easily and rapidly accessible to an applications program.